AMENDMENTS TO THE CLAIMS

This listing of the claims replaces all previous listings of the claims:

1. (currently amended) A method for holding up operands of R-unit registers for a minimum number of cycles until all prior updates have completed by comparing addresses of said R-unit registers in at least one queue and interlocking valid matches of said R-unit register addresses , the method comprising:

receiving a plurality of R-unit register addresses associated with a millicode architecture environment, said R-unit register addresses including at least one of a millicode general register and a millicode access register;

storing said R-unit register addresses in a plurality of queues;

accessing said queues;

comparing said R-unit register addresses;

determining matches between said R-unit register addresses; and

implementing one or more write-before-read interlocks after said determining produces a valid match, said one or more write-before-read interlocks being implemented until said comparing is no longer active, whereby operands of R-unit registers are updated during an operand prefetching period with a minimum number of cycles.

2. (currently amended) The method of claim 1 wherein one of said write-before-read interlocks causes a millicode read instruction not to execute.

3. (Original) The method of claim 1 wherein said plurality of queues includes a write queue, pre-write and a read queue.

4. (previously presented) The method of claim 3 wherein a bypass sends said R-unit register addresses when said read queue is empty.

5. (Original) The method of claim 3 wherein said comparing includes comparing said R-unit register addresses sent to said read queue against said R-unit register addresses sent to said

write queue.

6. (previously presented) The method in claim 3 wherein said determining includes matching valid R-unit register addresses of said write queue and said read queue.

7. (Original) The method in claim 3 wherein said determining includes matching said valid R-unit register addresses of said pre-write queue and said read queue.

8. (Canceled)

9. (currently amended) The method in claim 1 wherein said one or more write-before-read interlocks prevent millicode read instructions from being processed.

10. (previously presented) The method in claim 1 wherein a write queue accumulates said R-unit register addresses.

11. (previously presented) The method in claim 1 wherein said updating occurs when an SRAM receives the accumulated results from a write queue.

12. (currently amended) A system for holding up operands of R-unit registers for a minimum number of cycles until all prior updates have completed by comparing addresses of said R-unit registers in at least one queue and interlocking valid matches of said R-unit register addresses, the system comprising:

a plurality of queues for storing R-unit register addresses associated with a millicode architecture environment, said R-unit register addresses including at least one of a millicode general register and a millicode access register;

a comparator for comparing said R-unit register addresses in said plurality of queues and determining matches between said R-unit register addresses; and

a plurality of write-before-read interlocks that are implemented after valid matches of said R-unit register addresses are determined, said write-before-read interlocks being implemented until said comparing is no longer active, whereby operands of R-unit registers are updated during an operand prefetching period with a minim number of cycles.

13. (currently amended) The system of claim 12 wherein one of said write-before-read interlocks causes a millicode read instruction not to execute.

14. (Original) The system of claim 12 wherein said plurality of queues includes a write queue, a pre-write queue, and a read queue.

15. (previously presented) The system of claim 14 further comprising a bypass that sends said R-unit register addresses when said read queue is empty.

16. (Original) The system of claim 14 wherein said comparator compares said R-unit register addresses sent to said read queue against said R-unit register addresses sent to said write queue.

17. (Original) The system in claim 15 wherein said comparator determines said valid R-unit register address matches between said write queue and said read queue.

18. (Original) The system in claim 15 wherein said comparator determines said valid R-unit register address matches between said pre-write queue and said read queue.

19. (Canceled)

20. (currently amended) The system in claim 13 wherein said interlocks prevent millicode read instructions from being processed.

21. (currently amended) The system in claim 14 wherein said R-unit addresses are accumulated in said write queueit ~~is allowed for said R-unit register addresses to accumulate in said write queue~~.

22. (previously presented) The system in claim 14 wherein said R-unit registers are updated when an SRAM receives the accumulated results from said write queue.